Code-Besprechung 02.03.2021

Spyder-IDE Vorstellung

- Matlab-ähnliche Entwicklungsumgebung für Python
  - Editor
  - Command Window
  - Variable Editor
  - Figure Window
  - Profiler
- Lehrcode (+ Parallelisierung und symbolischer Berechnungen)
- Parallelisierung etwas langsamer als in Matlab (Implementierungsabhängig?)

Struktur des neuen Codes

- Idee
  1. Auf Grundstruktur in Matlab zwecks Abstimmung festlegen (in git)
  2. Umsetzung in Python
  3. Entscheidung für Matlab + .mex-files oder Python
- Vorschlag für Grundstruktur → siehe Matlab
- Ordnerstruktur: Funktionen/Klassen/Skripte/ReadME (best practice)
- Code in Matlab
  - prepareWorkspace
  - setupClass
  - solidClass
    - edof → lokale Element-Knotenfreiheitsgradtabelle
    - (fullEdof → lokale, komplette Element-Freiheitsgradtabelle)
    - (globalEdof → globale Element-Knotenfreiheitsgrade, bei mehr als einem continuumObjekt relevant)
    - globalFullEdof → globale, komplette Element-Freiheitsgradtabelle
    - (nodesDof → lokale komplette Knotenfreiheitsgradtabelle)
    - globalNodesDof →globale komplette Knotenfreiheitsgradtabelle
- dirichletClass
- runNewton, konzeptionelle Baustellen:
  - TODO: als Funktion? Wie Objekte einsammeln?
  - TODO: globale Freiheitsgrade zuordnen? datenManager?

TODOs

- git aufsetzen, Name?
- auf Struktur einigen
- automatisches Testen mittels Testskript von Beginn an: CI (matlabrunner) in git checken
- Basisklasse für Kontinuumselemente
- Superklasse für solids
- weitere Klassen und Elemente
  - solidThermoClass
  - solidElectroThermoClass
  - gemischte Elemente, statische Kondensation